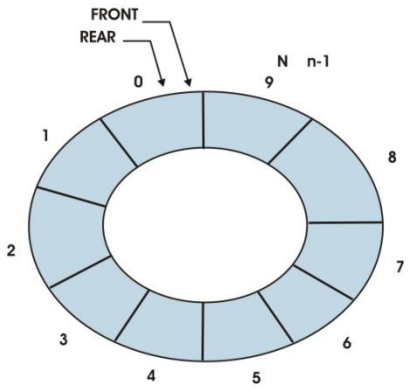
القوائم Lists

3. الطابور الدائري Circular queue:

1.3 التعريف Definition :

الطابور الدائري circular queue هو الطابور الذي فيه عملية الإدخال insertion لعنصر جديد new item في الموقع الأول first location للطابور إذا الموقع الأخير last location للطابور ممتلئ . الطابور الدائري يوجد له نقطة بداية لكن لا يوجد له نقطة نهاية.



نفـرض:

إذا لدينا طابور لــ n من العناصر ثم بعد إضافة العنصر في الدليل الأخير (n-1) الطابور يبدأ بالدليل 0 ، العنصر التالي يتم إدخاله في الموقع الأول first location للطابور ، وهذا غير ممكن في حالة الطابور الخطي linear queue . لهذا السبب يؤدي الطابور الخطي لسوء استغلال للذاكرة memory وهذا العيب في الطابور الخطي يتم التغلب عليه بواسطة الطابور الدائري.

2.3 العمليات التي تجرى على الطابور الدائري The operations of circular queue :

العمليات الرئيسية التي تنجز على الطابور الدائري circular queue :

1. enqueue(item) : إدخال insert عنصر في الطابور الدائري circular queue من مؤشر rear .
2. dequeue() : إلغاء delete وإرجاع قيمة العنصر الذي يأشر له المؤشر front .

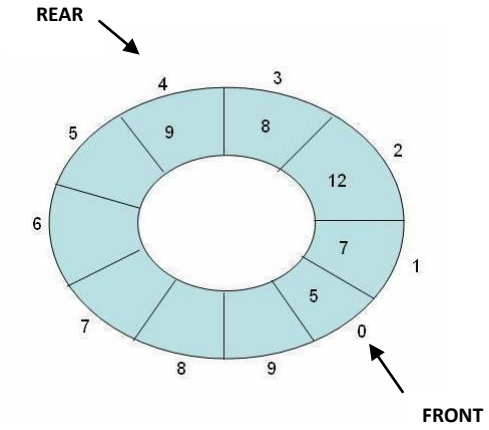
3.3 التطبيق للطابور الدائري Implementation of circular queue :

في أغلب لغات البرمجة العالية المستوى high level languages ، الطابور الدائري circular queue يمكن أن يطبق بسهولة.

```
CREATE  Q[n]  ,  FRONT ← -1  ,  REAR ←  -1
```

```
INSERT  ( Q[n] , REAR , x  )
IF  FRONT=0 & REAR = n-1 || FRONT=REAR+1  , THEN  "Queue Full"
IF  REAR = -1 & FRONT=-1  , THEN  FRONT ← 0
IF FRONT>0 & REAR = n-1 , THEN  REAR←0 , Q[REAR]  ←  x
ELSE REAR  ←  REAR + 1  ,  Q[REAR]  ←  x
END
```

```
DELETE  ( Q[n] , FRONT, REAR )
IF  FRONT = REAR , THEN  FRONT = REAR =-1  "Queue Empty"
IF  FRONT = n-1 & REAR < FRONT, THEN  FRONT = 0
ELSE FRONT  ←  FRONT + 1
END
```



4.3 برنامج بلغة السي لتطبيق الطابور الدائري بواسطة المصفوفة Code with C language for implementing a circular queue with an array :

العمليات على الطابور الدائري Operations on circular queue :

- **createqueue(q)** : لتكوين q كطابور فاضي queue empty .
- **enqueue(q,i)** : لإدخال عنصر i في q .
- **dequeue(q)** : للوصول وإزالة عنصر من الطابور queue q .
- **peek(q)** : للوصول إلى أول عنصر في الطابور queue q بدون إزالته.

تمثيل الطابور الدائري Circular queue Representation :

نستخدم الإعلانات التالية لتمثيل الطابور الدائري circular queue :

```
/*Define a queue of capacity 10*/
#define CAPACITY 10
typedef struct
{
    int front;
    int rear;
    int elements[CAPACITY];
} circular_queue;
circular_queue *q;
```

باستخدام هذه الإعلانات، نستطيع إنجاز العمليات المختلفة على الطابور الدائري circular queue .

تكوين الطابور الفاضي Creating an Empty Queue :

قبل أن نستخدم الطابور الدائري queue circular ، يجب أن يتم تكوينه . ويتم تكوين الطابور الدائري circular queue وذلك بتخصيص قيمة 1- لكل من المتغيرين front و rear . نلاحظ بأن المدى المسموح للمصفوفة هو من 0 إلى CAPACITY-1 .

```
void createqueue(circular_queue *q)
{
   q->front=q->rear=-1;
}
```

إنجاز عملية enqueue على الطابور الدائري Performing the enqueue operation on a circular queue :

```
void enqueue(circular_queue *q, int value)
{
  if(q.front==-1 && q.rear==-1)          /* Queue is empty */
    q->front=0;

  if ((q->rear == CAPACITY-1 && q->front == 0)||q->front == q->rear+1)
                                                  /*Queue is full*/
    {  printf("Queue is full");
          exit(0);
    }

  if ((q->rear == CAPACITY-1) && (q->front > 0))
     { q->rear=0;
       q->elements[q->rear]=value;
     }
  else
    { q->rear++;
      q->elements[q->rear]=value;
    }
}
```

إنجاز عملية dequeue على الطابور الدائري Performing the dequeue operation on a circular queue :

```
void dequeue(circular_queue *q)
 {
  if(q->front == q->rear)
    q->front=q->rear=-1;
   else
    if(q->front == CAPACITY-1 && q->rear < q->front)
      q->front=0;
    else
     q->front++;
 }
```

الوصول إلى العنصر الأول في الطابور الدائري Accessing to the first Element :

هناك قد تكون حالات نريد فيها الوصول إلى العنصر الأول في الطابور الدائري circular queue بدون إزالته من الطابور queue .

```
int peek(circular_queue *q)
{
    return(q->elements[q->front]);
}
```